HEWLETT PACKARD

Open Distributed Computing

The Overall Architecture

Version 1.0

NSA-93-019

Norman Kincl
Jack Armstrong

Architecture and Technology Application
Networked Systems Architecture

October, 1993

Open Distributed Computing is Hewlett-Packard's solution to cooperative computing in a multi-vendor, heterogeneous environment. Implementation in a customer enterprise information processing system should be guided by a set of architectural specifications we call the Overall Architecture.
This document defines the nature of the Overall Architecture and introduces the three main components—User Roles, the Hewlett-Packard Open Distributed Computing Architectural Framework and IT Processes.

Legal Notices

Trademarks

Printing History

The document was first printed in the United States of America on October 27, 1993.

This copy was printed in the United States of America on November 19, 1993.

Its master file name is I:\ATA\ARCH\OVERALL\PAPER.SAM

HEWLETT PACKARD
Networked Systems Architecture

Open Distributed Computing
The Overall Architecture

Version 1.0

NSA-93-019

Norman Kincl
Jack Armstrong

Open Distributed Computing is Hewlett-Packard's solution to cooperative computing in a multi-vendor, heterogeneous environment. Implementation of this infrastructure in a customer enterprise information processing system should be guided by a set of architectural specifications we call the Overall Architecture. This document defines the nature of the Overall Architecture and introduces the three main components—User Roles, the Hewlett-Packard Open Distributed Computing Architectural Framework and IT Processes.

1 Prologue

There are three crucial factors influencing modern business enterprises and their use of information processing technology—all involving change. First, businesses, and the world in which they operate, are being subjected to constant changes in markets, products, and customer expectations. Second, there has been a fundamental shift in the number of end users and their use of information technology. Finally, computing technology itself has advanced at a rapid rate.

The 1990s are arguably the fastest paced decade in human history. In business, science, and even politics, fast moving changes and events require increased efficiencies in time-to-market, service industry response rates and rapid adaptation to market fluctuations. To meet these challenges, the management process has to be accelerated as well. New business structures with fewer management reporting levels are allowing more rapid decisions to be made at lower levels in the organization. Businesses are finding that these new organizational structures increase the interdependence of entities and impose a critical need for broad access to information and tools.

This increased need for access to information has greatly increased the ranks of information technology users and the use of personal workstations—both personal computers and engineering workstations. Many firms have installed local area networks to allow teams of physically adjacent specialists and support staff to interact and share tools and information. With the introduction of modern networking technology, users of these systems now have access to an increasing range of services and information. With advances in networking technologies, many of the problems of physical connectivity were solved.

Unfortunately, by simply interconnecting existing systems, a daunting variety of confusing methods for accessing multiple applications and information bases was created. Historically, information processing systems have been developed by single vendors to solve a single problem, or at best a small set of closely related problems. These nonstandard proprietary systems provided barriers to interoperability between different applications and different vendors' platforms. Applications developed for specific types of users (e.g., engineers, technicians, data entry typists, secretaries) often appeared awkward and difficult to use by others. Finally, management and control of networks of heterogeneous systems are hopelessly complex. One solution to these obstacles—to start over and move dramatically to a new and consistent information technology—is prohibitively expensive. Not only would this incur the loss of investment in existing systems, but would require retraining of users and interruption of their normal work.

Recent advances in information processing technology—Reduced Instruction Set Computer (RISC) architecture, cost effective high volume data storage, high speed networks such as Asynchronous Transfer Mode (ATM), and the introduction of client/server architecture have all contributed to making many of the new access requirements practical. Advanced graphical user interfaces provide intuitive access to the systems we desire, but they require powerful computer systems and substantial memory to implement. Distributed applications impose greatly increased load on networks, but fortunately we are seeing better, faster, and more cost effective hardware solutions to meet these challenges.

Hewlett-Packard's Open Distributed Computing is designed to facilitate cooperative computing in a multi-vendor, heterogeneous environment that is based on open systems and standards. It is designed to allow flexibility in the computing environment, helping enterprises to adapt systems to changing needs.

In an Open Distributed Computing environment, users may transparently obtain services and resources from a network of computers behaving as a single integrated whole. Users may access the system through workstations, terminals, or personal computers integrated into this network. They have a consistent view of these services and resources regardless of location or implementation. This view is independent of the heritage of these services either as legacy applications or newer distributed, client/server applications.

Success in this endeavor will require an infrastructure that supports interoperability and distribution of diverse applications and collections of information. This infrastructure will be implemented on multi-vendor heterogeneous systems—encompassing mainframes, minicomputers, work stations, personal computers (some portable) and, increasingly, special purpose servers on an integrated network. Ideally, this infrastructure should be implemented entirely in common open standards, but investments in current applications, information bases, and training cannot be ignored or suddenly replaced. The rapidly emerging open standards must form the backbone of future systems, but migration and integration paths to these standards must be provided.

Hewlett-Packard's Open Distributed Computing is predicated on meeting the following major objectives:

- Provide users with enterprise wide access to information and information processing facilities, without exposing the complexities of the underlying systems.
- Integrate heterogeneous hardware, operating system, and network platforms.
- Make pervasive use of open standards.
- Implement client/server architectures, while utilizing and enhancing existing legacy systems.
- Allow administrators to manage these computing environments at the highest level of abstraction commensurate with the systems and devices being managed.

Meeting these objectives in a specific customer enterprise computing environment will require a rigorous design methodology and complete specifications that guide the implementation, management and use of that environment.

2 Introduction to the Overall Architecture

The Overall Architecture of an enterprise information processing system encompasses all aspects of that system: structure, functionality, environment, and operational processes. To define better what an Overall Architecture is, we must first define some basic terminology.

- *System* — "a set of different [components] so connected or related as to perform a unique function not performable by the [components] alone" [Rechtin91]. Each of the components may be a complex system in its own right.
- *Information Technology* (IT) *System* — the enterprise-wide information processing system. The purpose of an IT System is to provide the technological support to an enterprise to meet its information processing needs. Its components include computers, networks, applications, data bases and support services.
- *Computer System* — an individual computer that, together with the networks, applications and other components, forms the IT System.
- *System Architecture* — specification of "the structure of the system" [Rechtin91]. Thus, the System Architecture specifies the complete structure of the IT System.
- *Overall Architecture* — "the structure not only of the system, but of its functions, the environment within which it will live, and the process by which it will be operated" [Rechtin91].

Thus the overall architecture should address more than just the structure of the various components making up the IT System. While it is critical to understand the various pieces of the IT System and how they fit together, this is not sufficient. To build and operate an IT System that successfully addresses the business objectives of an enterprise, the overall architecture of the system must be understood.

2.1 Basic Model

The Open Distributed Computing Architecture deals with three major areas. The first is concerned with the people involved with the system. In particular, it looks at the major roles that users of the system play, and the interactions between these roles. The second area of focus is the technology that comprises the system. Here we answer questions such as "what are the components that make up the system and how do they relate?" Finally, the Architecture covers the IT processes. Here the focus is on processes used to build, maintain, administer and evolve the system. These three areas fit within the overall architecture of the system—the IT system is part of a broader environment that includes the whole enterprise and its partners.

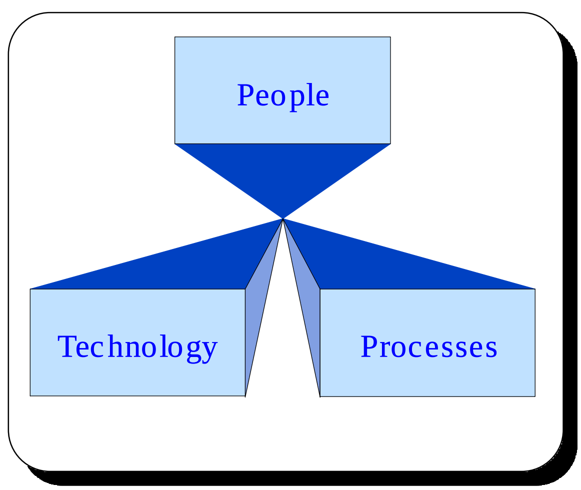

Figure 1: Basic Model

The following sections cover each of the areas of the architecture separately.

3 People — User Roles

People interact with the information system in many different ways, depending on their role (or roles) within the enterprise. In a small enterprise, one person may be active in all the roles. In a very large environment, each of the roles may consist of many people, or even complex organizations. The roles categorize the types of interactions with the system.

3.1 End User

The primary reason for any information processing technology is augmenting the participants' skills needed for the business activities necessary to operate an enterprise. End users regard the system as a tool to support their daily business activities. Typically, systems are funded by the end users or on their behalf. Successful use of a system by these users is the ultimate measure of any investment in information processing technology.

The need for faster answers to complex problems or simply the elimination of soul-destroying repetitive tasks has not changed since the advent of the first computers. What *has* changed is the ability to apply this technology to the problems of decision making and communications on a global scale. Open Distributed Computing strives to extend these technological advances to the largest possible audience of end users in ways that best augment their ability to contribute to the fundamental business of the enterprise.

3.1.1 Enterprise Customers and Partners

In some businesses, there is an additional user role—that of users external to the enterprise. These include the enterprise's customers and business partners who have direct access to the enterprise IT system. For example, a bank customer using an Automatic Teller Machine or on-line account access is a user of the bank's information system. A company may give its preferred suppliers direct access to portions of its IT system. In an information utility enterprise, the vast majority of end users will be in this category.

3.2 Builder

Systems that provide access to distributed tools and information simplify the end user's life, but seriously complicate that of system builders. The very act of masking the underlying complexities of these systems from the user requires dealing with a level of complexity not present in monolithic systems. Builders themselves require sophisticated tools, and a runtime environment that provides a rich set of services to support distributed applications.

The builder role involves both building and evolving the system. Two separate concepts are involved here. First, component providers need to provide the components from which the system is built. Second, a system integrator needs to architect and engineer the IT system.

3.2.1 Component Provider

A component provider creates components that will be part of the system. These components can be simple individual pieces of hardware, or complex systems on their own (e.g., computer systems).

3.2.1.1 Component Developer

Normally, a component developer builds components (hardware or software) that are application independent. This work is usually not done by the enterprise. Except for applications, most components are purchased..

3.2.1.2 Application Developer

The application developer is also a component provider. Applications are specific to the enterprise business requirements. There is a trend towards buying off-the-shelf applications where feasible, but exceptions are made for unique applications or those that provide competitive advantage.

3.2.2 System Integrator

The system integrator combines the various components into a system according to a defined architecture. The system integrator may need to write some integration and interfacing software to accomplish this task. Increasingly, this function is outsourced.

3.3 Administrator

Administrators watch over the system to assure that it is meeting the Quality of Service goals dictated by the business needs of the enterprise. Administrators need full control within their domain, or sphere of responsibility, with as little interference from the rest of the network as possible. As more systems are joined by networks, the configuration, management, and control of multi-vendor networks become enormously complex.

The activities of administrators fall into four major categories: end user services; operations; administration; and maintenance and installation.

3.3.1 End User Services

Administrators are responsible for assisting the end users and ensuring that they derive maximum benefit from the available systems. These activities typically include

- operate help desk services
- provide on-site support
- provide end user consulting
- track end user problems
- provide end user training
- monitor Service Level Agreements

3.3.2 Operations

Operators work to guarantee smooth daily operation of all components of the system. This typically involves managing

- operational problems
- problem escalation
- output (e.g., central printers)
- storage resources (e.g., tapes)
- job control and flow
- console operations

and monitoring

- performance
- security

3.3.3 Administration

System Administration encompasses the high-level functions required to implement strategic business policy and to manage system resources accordingly. Typical functions are management of

- system configuration/change
- security configuration
- user access/options
- system accounting

3.3.4 Maintenance and Installation

Maintenance and Installation provides for the physical aspects of the system. These activities cannot be totally centralized—they require on-site support. Example functions are

- installing hardware
- installing and maintaining cable infrastructure
- maintaining system components
- ordering and stocking supplies

3.4 Strategic Planning

Strategic planning is the link between the IT system and the business. Strategic planning will

- define the information infrastructure
- define IT architectural strategy and direction
- coordinate and support the work of the builders and administrators

Strategic planning activities include

- define the Information Systems strategy
- define the Architecture
- define IT processes
- negotiate Service Level Agreements
- implement planning
- evaluate projects
- manage cost control
- perform asset and budget planning

3.5 Common Role Interactions

Figure 2 on page 7 describes some common interactions between the various user roles and the IT system. This is not meant to be a complete set, but rather an example of some of the most common types of interactions that occur.

It is important to understand these interactions for two reasons. First, they define some of the requirements placed upon the system. This is primarily seen by looking at the details of the interactions that the user roles have with the IT system. Second, these interactions have an impact on the processes that surround the system. We address processes further in the section "Process" on page 16.

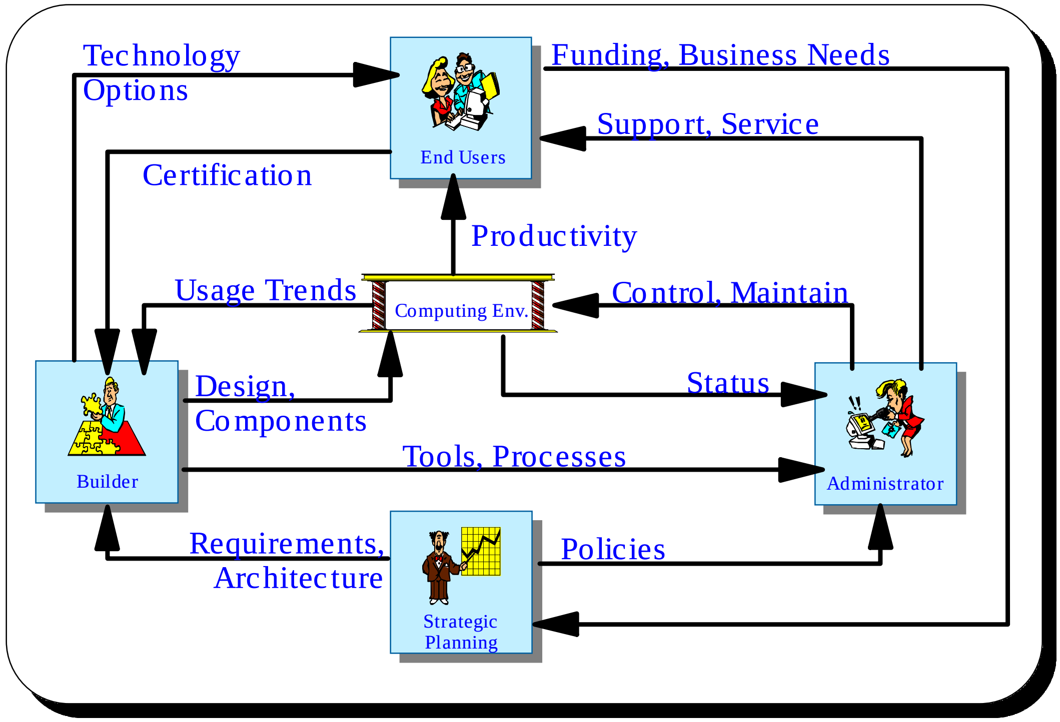

Figure 2: Common User Role Interactions

4 Technology — The System Architecture

The implementation of an information processing system should be done according to a System Architecture designed to meet the needs of the enterprise. The goal is to provide open distributed computing in a heterogeneous environment. Our starting point for this process is Hewlett-Packard's Open Distributed Computing Architectural Framework. This framework establishes the context, general concepts, requirements, relationships, and terminology to be used in subsequent levels of architectural design and resulting systems. Within this framework, and according to its guidelines, detailed architectural designs may be developed and integrated with other related architectures. The framework will provide the vision, direction, and master template for seamless inclusion of many existing systems and heterogeneous platforms, specifying the standards required for interoperability and support of distribution and heterogeneity.

Since every enterprise has unique needs that need to be met by an information system, we introduce here another concept—reference architectures. A reference architecture provides an example system architecture useful for particular domains and is derived from the Architectural Framework.

4.1 The Framework

The Open Distributed Computing Architecture is built upon an architectural framework. The architectural framework is a high-level model and a set of guidelines for building system architectures. As shown in figure 3 on page 8, this framework is not specific to any business, technology or user domain, but is derived from domain-independent characteristics. The requirements for open standards, heterogeneity and distribution have been used to create the Open Distributed Computing Architectural Framework.

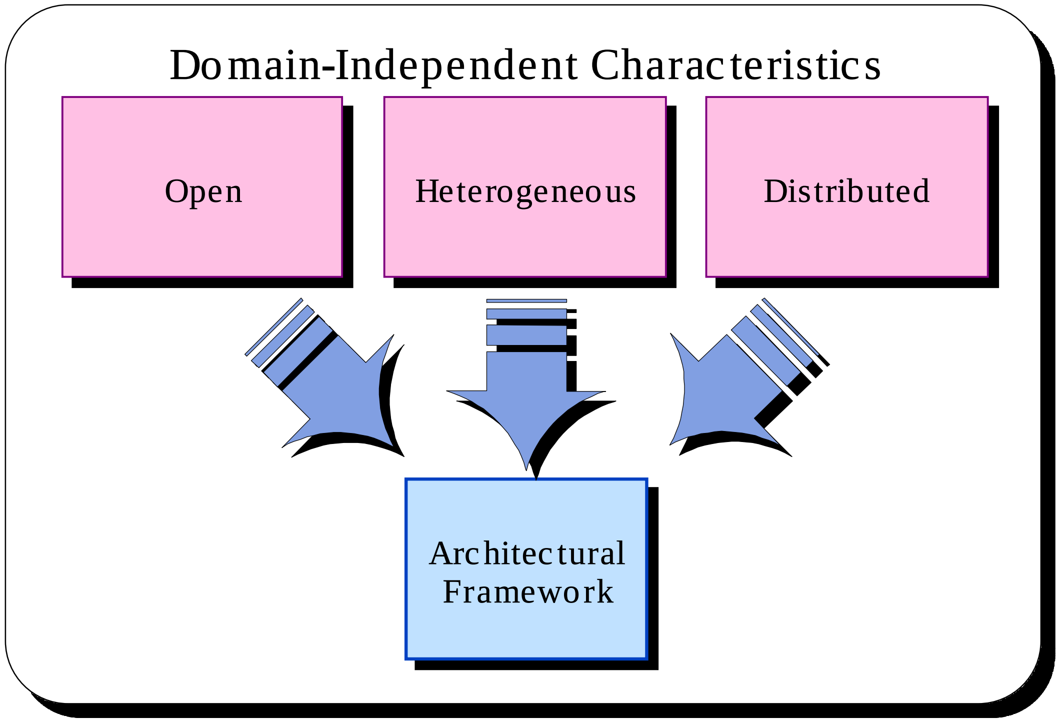

Figure 3: Framework Derivation

A key element of Hewlett-Packard's Open Distributed Computing is an emphasis on adoption of and adherence to standards. The acceptance and utilization of standards protect major investments in applications and people. Standards increase opportunities for interoperability and use of applications by a broader range of users, and will yield an increased return on investment. Use of standards increases opportunity for vendor independence and lower costs.

There is a broad spectrum of standards—ranging from private, proprietary *de facto* standards (such as MS-DOS from Microsoft), to public proprietary standards (such as Sun Microsystem's NFS), to open consortia standards that are not controlled by a single vendor (such as DCE, from OSF), and finally to open *de jure* international standards (as from ISO and CCITT). Hewlett-Packard has a long-standing commitment to the use of open international standards wherever possible, and has participated in the formation of and contribution to these standards. Where these standards are lacking, Hewlett-Packard has worked for the development of *consortia* standards for many years—for example, as active participant in X/Open and founding member of the OSF, OMG, and COSE. As a matter of corporate policy, Hewlett-Packard has worked to achieve acceptance of our own proprietary standards as open, public standards and will continue to do so. Our record of success in providing many of the standards for *de jure* networking standards and contributions to major consortia attests to this commitment.

As important as it is to strive for open, vendor independent standards wherever possible, *de facto* standards are still preferable to ad hoc proprietary solutions. *De facto* standards achieved their status by establishing a large installed base and cannot be ignored.

As the desire to interconnect these systems is being met with new networking technologies, the era of single vendor systems is ending. We have moved irreversibly into enterprise-wide systems that are not only widely distributed, but include a broad range of vendors, applications, data base systems, and data communications suppliers. This heterogeneous characteristic of the architecture includes both hardware and software.

Distribution of systems may be a result of business requirements, or may provide advantages in performance and reliability. Distribution of both the information and processing has become a pervasive requirement in modern computing.

4.1.1 Reference Architectures

The architectural framework is useful in structuring the solution, but to address the design of a specific distributed system requires the greater detail that is found in a Reference Architecture. Reference architectures add domain-specific characteristics to the architectural framework. A reference architecture provides additional specifications and details to the framework. It can also be thought of as an example system architecture useful for a particular domain.

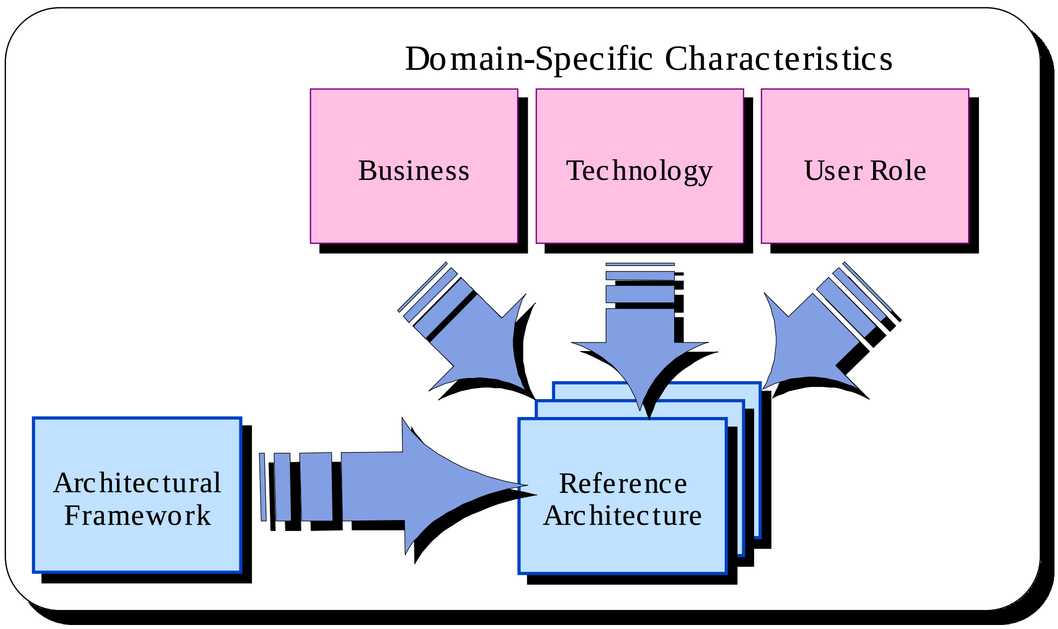

Figure 4: Reference Architectures

Figure 4 on this page shows the three types of domain-specific characteristics used to create reference architectures—business, technology and user role. A reference architecture could be based on just one, two or all of these types of domain-specific characteristics. The business characteristics are defined by business models and typical information system requirements for a particular business segment. Business segments are areas such as finance, telecommunications, manufacturing, medical, etc. Businesses are best characterized in term of business activities—e.g., insurance claims processing, opening or closing customer accounts, etc. The technology characteristics provide a technology model and requirements necessary to achieve some widely used functionality. For example, a technology model may be used to describe highly available systems. A reference architecture based on this model would provide the necessary information required to create a highly-available distributed system.

The user role characteristics describe the model and requirements of users in a particular role. For example, administrators will have a very different view of the system than software developers. A reference architecture for administrators would give an example of an architecture for supporting their work.

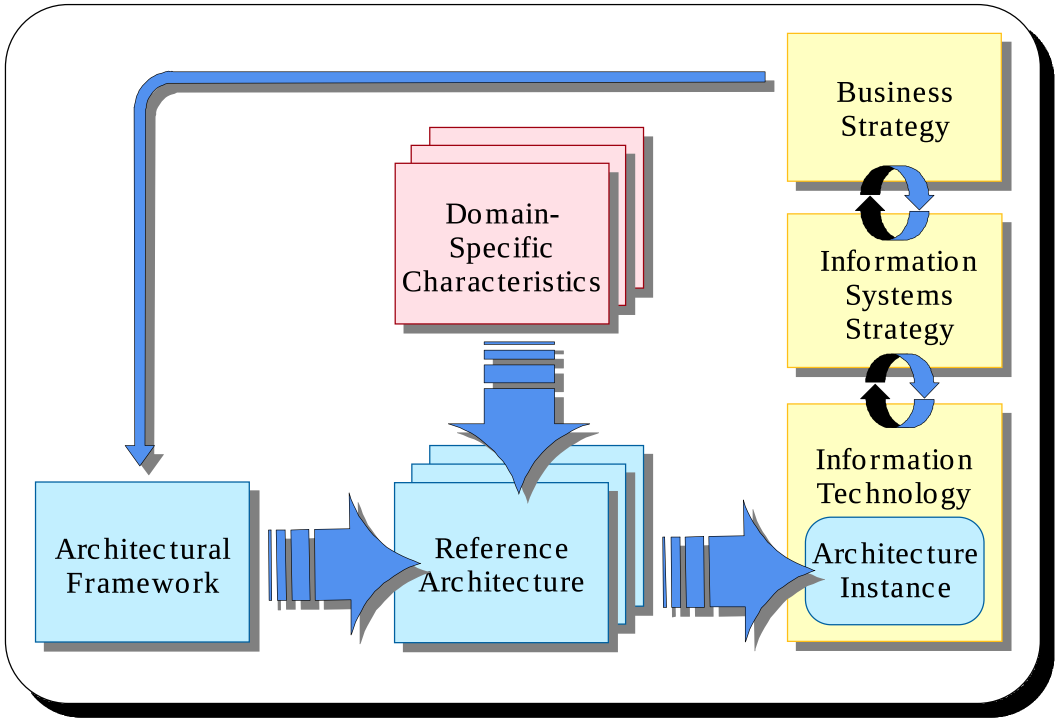

Figure 5: Enterprise Perspective

4.1.2 Enterprise Perspective

Enterprises are interested in a system architecture that will meet their business needs. Though reference architectures will address the particular business segment of the industry, they cannot be expected to address the particular needs of any one enterprise. Reference architectures can, however, be used as a base from which to build information technology systems that meet such business needs.

Figure 5 on this page illustrates this process. The enterprise's business strategy drives the information systems strategy by dictating what information is needed. The information systems strategy defines the type and nature of information processing required to meet the business strategy. The information technology provides the technological support to meet the information systems strategy. Advances in information technology will evolve the information systems strategy. This may then become one of the factors in the evolution of the business strategy.

The business strategy is based on value judgments. These value judgments are used to define and refine the domain-independent characteristics used by the framework.

A system architecture defines the information technology system. The enterprise can base the architecture of the information technology system on one or more Open Distributed Computing reference architectures. Reference architectures provide a rich starting point that facilitates creating an open, heterogeneous, distributed computing system. These reference architectures are combined and tailored to build a system architecture that meets the specific information processing needs of the enterprise. This process facilitates the development of an enterprise information system that supports distributed, heterogeneous components with open systems technologies.

4.1.3 Product Architectures

A well-architected information processing system will allow the inclusion of products from different vendors. The products that are the components of the system have their own internal

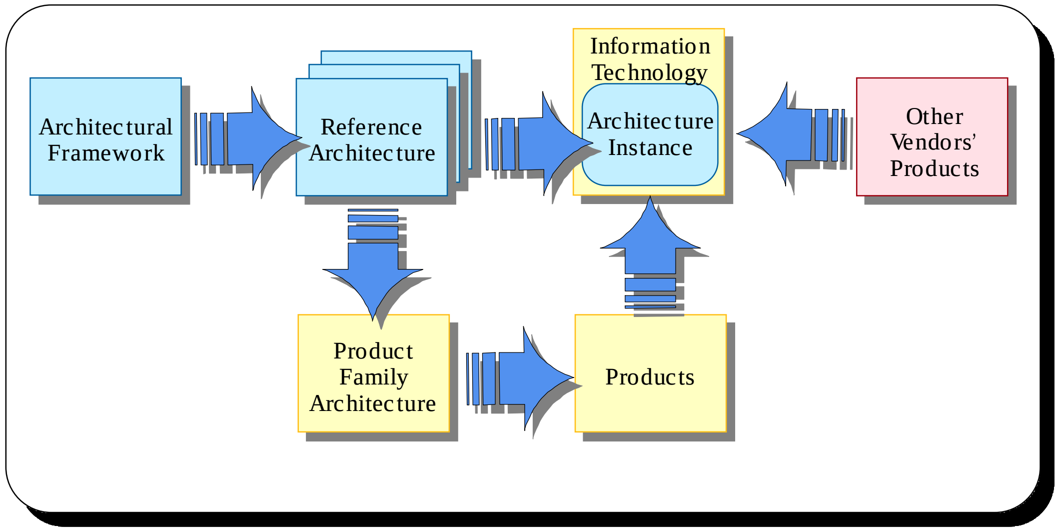

Figure 6: Product Architecture Perspective

product architecture. An enterprise should be interested in the product architectures of those products that make up their system. A product whose architecture is based on Open Distributed Computing will have a better fit within the system than other products. Figure 6 on this page shows how product architectures relate to reference architectures.

There are three advantages to be gained by purchasing products based on Open Distributed Computing reference architectures. First, the products will be able to participate in open, heterogeneous and distributed systems. These are basic characteristics that the reference architectures derive from the framework. Use of the reference architectures will clarify requirements for such systems and simplify combining the components into a total systems solution.

Second, the products address the specific needs of the domains used by the reference architecture. The reference architectures are not meant to replace customer input. Rather, they take customer input and use it to shape a system architecture.

Third, the products are designed to be components of a larger system. Though they can operate on their own, they are designed to integrate easily with other products into a larger system.

4.2 The Framework Model

The model used to describe the Open Distributed Computing Architectural Framework is the cube shown in figure 7 on page 12. The front face of the cube depicts a set of modules. These modules describe functionality required in an Open Distributed Computing system, depicting the first-level decomposition of the system. These modules are

- base platform
- distribution services
- information management services
- application cooperation services
- utility services
- applications
- user interface services

These are further defined below.

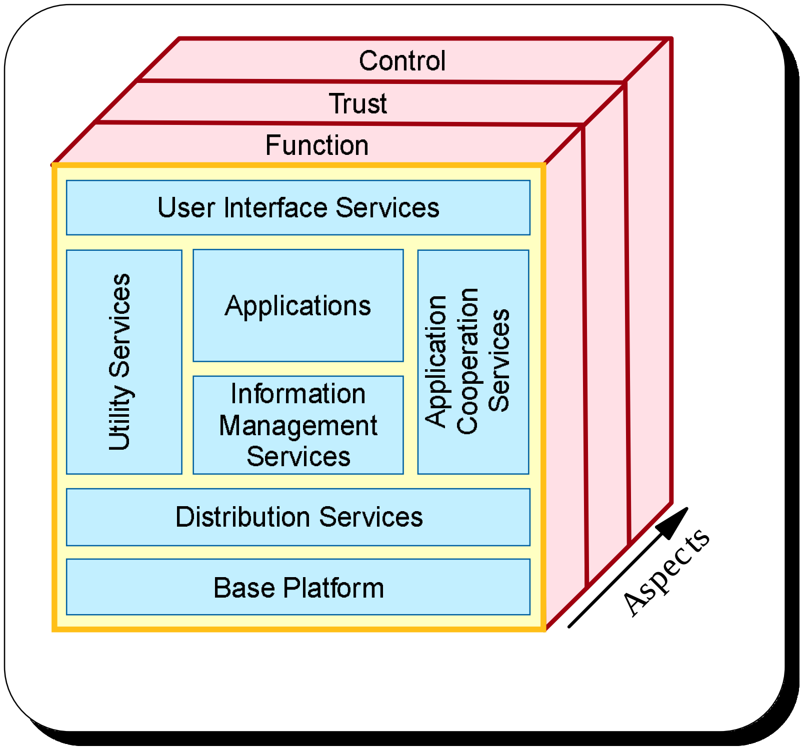

Figure 7: The Framework Model

The third dimension, or depth, of the cube depicts the aspects of the modules. These are fundamental characteristics required of the modules. The aspects are

- function
- trust
- control

These are further defined in the section "Aspects" on page 15.

4.3 Modules

The modules provide the first-level functional decomposition of the system. Each module is responsible for providing specific functionality to the system. This provides a functional decomposition of the entire system—not of any one computer system or node. Not all nodes in the system need to contain all the modules. Some of the functions are required to be present in every node while others are required to exist someplace in the IT system but need not be present in all nodes. Finally, some functions are optional, based on the needs of the enterprise.

4.3.1 Base Platform

The base platform contains the basic processing components of the IT system. By definition, every computer system in the IT system will always have base platform functionality. This includes the system hardware, operating system and peripherals that make up the computer systems within the IT system. To the rest of the IT system, a computer system appears as a single component—even if it is composed of multiple CPUs. The base platform includes characteristics such as:

- processor types (e.g., PA-RISC or Intel 80x86)
- processor configuration (e.g., single CPU, Symmetric Multi-Processor)

- system bus (e.g., EISA)
- operating system, including the interface (e.g., IEEE Posix 1003.1) as well as any other special characteristics
- peripherals, including common system peripherals (e.g., disk drives) as well as special-purpose peripherals (e.g., automatic teller machines)
- network infrastructure, both the cable plant and the hardware used to provide the network (e.g., modems, repeaters, bridges, routers)
- instruments used to facilitate the operation of the IT system

4.3.2 Distribution Services

Distribution services provide the functionality that links multiple, separate computer systems into a distributed system. They include

- network services (e.g., TCP/IP, OSI network protocols)
- invocation services (e.g., remote procedure call, message passing)
- location services (e.g., trader, directory service)
- security services (e.g., authentication service, authorization service)
- system coordination services (e.g., time synchronization service)

4.3.3 Information Management Services

Information Management Services are a set of services that organize, store and retrieve information. For example

- file systems
- access methods (e.g., indexed sequential files)
- database systems (e.g., relational DBMS, object-oriented DBMS)
- document stores
- information semantics (e.g., SGML)

4.3.4 Application Cooperation Services

Application cooperation services provide facilities that enable applications to cooperate with each other to solve a common business activity. This module builds upon the communications provided by the distribution services and the data sharing provided by the information management services. The module adds the coordination functionality required to enable applications to cooperate. Typical services include

- transaction processing manager (e.g., X/Open DTP standard)
- enhanced messaging services (e.g., event management, reliable message queues)
- object request broker (e.g. OMG's CORBA)
- workflow management
- agents
- encapsulation facilities

Some of these services require the applications to be written with a knowledge that they will cooperate through the service. Other services allow the application to cooperate without *a priori* knowledge.

4.3.5 Applications

Applications supply the business logic required to support end user tasks. Applications fall into three major categories. They can deal with

- specific business activities of an enterprise (e.g., a loan-processing application for a bank or a factory-automation application for a manufacturer). Though some of these applications may have commonalty across business domains, they are typically specific to one domain. Often, some of these applications will provide a competitive advantage.
- general business processes frequently required by many types of businesses (e.g. human resource management, asset control, accounts receivable). There is generally little value in creating custom applications to address these needs and they are typically purchased from a software provider.
- personal or group productivity. Some of these may be standardized across an enterprise (e.g., a corporate standard word processor), whereas others may remain the personal choice of the individual or group using them (e.g., a time management application). There is rarely value in creating custom productivity applications.

Though in most cases customers want to buy rather then build applications, they want these applications to work together. While general business applications are not specific to the customer's environment, they may need to access information that is specific to the customer's business.

4.3.6 Utility Services

Utility services provide general system-related functionality for applications. The focus of the applications module is to support the end user business tasks. Utility services are applications or libraries that facilitate using the system or help it function. For example

- spooling system (e.g., print, batch job)
- resource accounting service (e.g., software licensing service)
- language run-time systems (e.g., Smalltalk runtime)
- common libraries (e.g., math library, sort routine)

4.3.7 User Interface Services

User interface services provide the link between the system and the users of the system. These services provide the functionality to present information to, as well as acquire information from, the user. Presenting information may include

- graphical user interfaces
- desktop management
- multi-media output
- page description languages

Data can enter the system from multiple sources. Ideally, data is automatically captured when it is generated. For example

- point of sale terminals
- bar-code readers
- directly-connected analytical instruments

The functionality to deal with these types of input is part of the user interface services. As voice and handwriting recognition systems are developed, the user interface services will provide the functionality to deal with these new types of input.

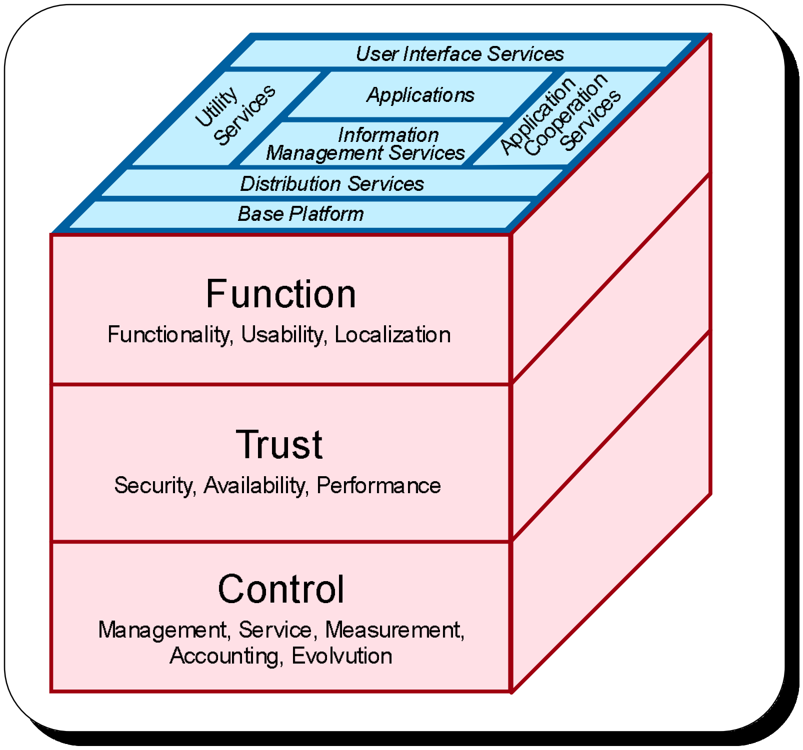

Figure 8: The Framework Aspects

4.4 Aspects

The aspects are the fundamental characteristics of the modules that make up the system. As we look at each module of the system, we need to examine each of its aspects. Although in some environments some of the aspects may take on more or less importance, they all must be considered for each component of the system. The aspects are shown in figure 8 on this page.

4.4.1 Function

The function aspect answers the question "what is the purpose of this component?" It deals with each component's responsibility to the system as a whole. Directly or indirectly, each component needs to help the end users do their job. Another way of looking at this is that it answers the question "how does it help get the job done?" The function aspect of a component is used to place the component in the appropriate module. (This functionality aspect was discussed and used to define the modules of the Architectural Framework).

The function aspect defines several characteristics of the components. First it defines the interfaces to the components. The interfaces to a component provide the necessary information to allow other components to use or access it. The interface definition needs to include not just the syntax (what does the interface looks like?), but also the semantics (how does it behave?).

Second, the function aspect deals with the usability of the component. The usability needs to be seen from the perspective of multiple user roles. It is not sufficient to look at the end user's perspective of usability (though this is a critical perspective). For example, the ease of programming to a particular Application Program Interface (API) is also part of the component's usability.

The third part of the function aspect is the concept of localization—can users tailor the component to local natural languages? Even though an application may have all the required functionality, if it presents its information only in a language foreign to the user, it will not meet the business goals.

4.4.2 Trust

The trust aspect answers the question of "why should you trust your business processes to the system?" Unless the components that make up the IT system meet the required trust level, people will not use the system. Trust has three major parts to it—security, availability and performance.

The security part of the trust aspect deals with the system's ability to adequately protect the information in the system. Before anyone will trust a system, they need to know that their use of the system is secure, that their confidential information remains confidential, and that they have access to the information that they need. Security also helps ensure that the system will be used and not abused.

The second part of the trust aspect is availability. If the system does not meet users' availability requirements, they will not be able to use it. Even though it does exactly what they need to meet their business requirements, it needs to be available *when* they need it. Availability includes topics such as recovery (after failures) and high availability (including fault tolerance).

The final part of the trust aspect is performance. In some environments, not meeting a certain performance level is equivalent to failure. This defines a mandatory level of performance. For example, a real-time video link has certain required throughput and latency requirements. If they are not met, real-time video is impossible. Similarly, controlling a complex piece of equipment such as a nuclear reactor has stringent response time requirements. Not meeting them could be catastrophic.

In addition to the mandatory performance, there is the broader issue of *acceptable* performance. Consider a system that does exactly what the users want, meets their security concerns, and is available whenever they need it. However, if it impedes work to an aggravating degree or the task is faster to accomplish manually, they are unlikely to use the system.

4.4.3 Control

The last aspect is control. This answers the question "how can you control the components so that they behave correctly?" It also addresses evolution of the system—"can the system evolve to meet the changing business needs?" This aspect of the components should not be confused with the management platform and applications used to manage the system. Rather, it is those aspects of a component that allows it to be managed. There are three parts to this aspect: control, measurement, and evolution.

Controlling a component requires manageability and serviceability. Manageability involves controlling the component or system under normal situations. Included here are management interfaces to components. These define the interface into a component used to control it. Serviceability deals with being able to fix things when they break (or protect them from breaking). Must someone be physically present to fix something or can it be fixed remotely? Can an impending failure be detected? Is preventive maintenance required, and if so, can it be accomplished automatically and remotely?

There are two types of measurements that must be made on components. The first deals with the performance and state of the component. This is closely related to both manageability and serviceability. The second type of measurement involves the accountability of the component. This involves accounting for its usage. Typically, enterprises charge costs to the users of a service. The granularity of what to charge for depends on the particular business guidelines of the enterprise. This granularity will dictate which components need to account for their utilization.

Finally, the control aspect covers the ability of the component to evolve. Over time, the business strategy of the enterprise will change. This, together with the availability of new technology, will drive the evolution of the IT system. The more adaptable a component is, the longer a life it will have in the continually-evolving system.

5 Process — The Question of "How to?"

A process defines the actions required to meet a particular goal. The business and Information System models define the business processes of an enterprise. These are the processes that meet the business goals of the enterprise. Example business processes involve manufacturing, customer support and new product development. As enterprises seek to improve the quality of the product and service that they deliver, these business processes are being defined and formalized. The ISO 9000 standards are providing a strong push in this direction.

Many of the business processes use the IT system to implement or support them. This is another way of viewing the primary purpose of the IT system—supporting the end user.

5.1 IT Processes

The IT system is involved with processes in another way. Figure 1 on page 3 introduces processes as part of the overall system architecture. In a traditional system, these would be the processes required to operate the system. However, IT systems are constantly evolving and adapting to the changing needs of the enterprise. Because of this, Open Distributed Computing broadens the scope of the processes to include those required to design, build and run the system. Though there is a strong link between the processes and user roles, all processes may involve people in different user roles.

5.1.1 Designing IT Systems

The process for designing a distributed system starts with a model of an enterprise's Information System. Taking this, together with the enterprise's business policies and guidelines, the process creates an architecture and engineering design for the system. Though the initial design of the IT system is important, the focus of the process definition needs to be on the continual evolution of the system. Hewlett-Packard's IT Planning Methodology is an example of a process for designing a distributed system.

Some of the processes that are used to design the IT system are

- collect user needs
- define IT policies and procedures
- create the architecture
- develop a design
- evolve the system architecture as dictated by business or technology changes

5.1.2 Building IT Systems

The process to build a system involves two separate parts. The first is a process for building components of the system. The many different software methodologies available address this. The second part is a process for integrating components into a system. As with the designing of the IT system, the processes for building the system need a strong focus on adding on to an existing system, rather than building a system from scratch.

Example processes used to build an IT system are

- software development

- documentation
- scenario prototyping
- system integration
- configuration
- acceptance testing and certification

5.1.3 Running IT Systems

The final set of IT Processes deal with keeping the system running. These include all the maintenance and administration processes. Examples are

- fault isolation, recovery and repair
- service activation
- preventive maintenance
- reconfiguration
- usage trend analysis

5.2 Understanding IT Processes

As we seek to understand any process, we can look at four questions—what? who? how? and why? Answers to these questions give us knowledge in two different areas—the tasks of the process and the context of the process. Knowledge of the tasks involved in a process gives us an understanding of the steps required to achieve a goal. Proper knowledge of the steps allows a formal definition of the process.

The context of the process places the process within a larger scope. Understanding the process' context involves understanding the goal to be achieved. This includes knowing the value, justification and risks of the goal, as well as how this goal is related to other goals. The process' context also involves knowing, and being able to measure, success and failure. The context knowledge has a strong link to the user roles through which people interact with the system. A knowledge of the process context is vital in optimizing the process.

5.2.1 Defining IT Processes

Most processes are not single monolithic steps. Rather, they can be decomposed into parts that together accomplish the intended goal. We see processes as composed of activities, which in turn are composed of tasks.[1] The purpose of defining the process is to document the activities and tasks that form the process. To be useful in later stages, the process needs to be defined using a formal method. Each task, activity and process needs the following defined:

- Startup criteria—what causes this task to be started?
- Inputs—what input is provided to us to accomplish the task? Is all the input always available before the task can start?
- Tools—what tools are available to help accomplish this task? Are the tools required or optional?
- Outputs—what output does the task generate? Is the output used as input to another task?
- Completion criteria—how do we know that we have finished the task?
- Metrics—what do we measure to know how well we have accomplished the task?

[1] This is a somewhat arbitrary breakdown. In reality, processes can be recursively decomposed into smaller and smaller steps. At some point, further decomposition becomes counter-productive.

In addition to the above, activities need to have a description of the tasks, and their relationships, that together form the activity. Similarly, the process needs a description of the activities that form it.

Having the processes defined in these terms allows us to go to the next step—optimizing the processes.

5.2.2 Improve IT Processes

The goal of understanding and defining the IT processes is to be able to improve them. Improvement takes several different (simultaneous) approaches. For the defined tasks, activities and processes, we need to

- eliminate those that are unnecessary
- automate those that don't require intervention from a person
- allow them to occur in parallel

Redefining the processes can accomplish some of this improvement. This requires a good understanding of the context of the process. This type of improvement can be done by examining the task and activity definitions. With an understanding of the process context, it is possible to eliminate some tasks because they do not substantially contribute to the desired goal.

A more thorough improvement of the processes involves understanding the effectiveness of the processes as defined. The effectiveness is evaluated using the metrics that are part of every task's definition.

However, the most thorough improvement of the processes requires viewing the overall architecture of the system. We cannot look just at the processes but also need to examine the user roles and technology that make up the system. To improve efficiency of the system, it may be advantageous to redefine user roles or change some of the technology in the system. Such a holistic view requires an iterative evolution of the system. Though not easy, the results are a smoothly running system that successfully addresses the business objectives of the enterprise.

6 Glossary

Architectural Framework	A high-level model and a set of guidelines for building *system architectures*.
Architecture Instance	The system architecture for the Information Technology system of a particular enterprise or part of an enterprise. "A formal specification of the way the … computing solution will be organized and executed." [Gartner93]
Business Model	A description of an enterprise. It includes business goals, organization and procedures.
Business Policies & Guidelines	A body of rules by which an enterprise is operated. Policies are usually mandatory whereas guidelines are suggestive.
Client/Server Computing	"A processing model in which a single application is partitioned between multiple processors (front-end and back-end) and the processors cooperate (transparent to the end user) to complete the processing as a single unified task. Client/Server computing is endlessly recursive; in turn, servers can become clients and request services of other servers on the network." [Boar93]
Domain	A collection of enterprises, technologies or *user roles* sharing common characteristics. Domains can be based on one or more types of characteristics (business, technology, or user-role).
Information Systems (IS)	The processing of information required to implement the business functions of an enterprise.
Information Technology (IT)	The technological support to an enterprise to meet its *Information System* needs. *Information Technology* includes computers, networks, applications, data bases and support services.
Infrastructure	An implementation of a fundamental and common set of components in the *system*.
IT Policies & Guidelines	Rules for building the IT system. These are derived from requirements or can come from *Business Policies and Guidelines*, external regulations, or standards that must be followed in a particular *domain*. Policies are usually mandatory whereas guidelines are suggestive.
Methodology	A body of methods and rules to be followed as a structured discipline.
Modules	A set of components or *subsystems* related through providing similar functionalities.
Overall Architecture	"Includes the structure not only of the *system*, but of its functions, the environment within which it will live, and the process by which it will be built and operated." [Rechtin91]
Process	A series of actions or operations used to achieve a particular result.
Reference Architecture	An example *system architecture* addressing a specific *domain*.
Subsystem	A proper subset of a system, which is a *system* in its own right.
System	"A set of different [components] so connected or related as to perform a unique function not performable by the [components] alone." [Rechtin91]

System Architecture	"The structure of the *system*." [Rechtin91]
System Design	Full specification of an *Information Technology system* for a particular enterprise. Based on an *Architecture Instance*, it specifies all product and version information required as well as topology information.
User Role	A classification of the different ways users interact with the *system*.

7 References

[Boar93] Bernard H. Boar, *Implementing Client/Server Computing*, McGraw-Hill Inc., 1993.

[Gartner93] W. Melling, "Strategies for Architectural Transition", Midrange Computing Strategies R-286-114, Gartner Group, March 1993.

[Rechtin91] Eberhardt Rechtin, *Systems Architecting: Creating & Building Complex Systems*, Prentice Hall, 1991.

8 Acknowledgments

Numerous people assisted with the writing of this paper. Contributions came as discussions, direct input, and reviews of draft copies. In particular, the people working on the Information Technology Strategy project provided valuable input. That project includes people from the Professional Services Division as well as HP Consultants. This assured that there was a direct link with customer requirements.